XP-002089719

P.D. 15-05-98
p 6-18 = (13)

2 Overview

The HAVi Architecture specifies a set of Application Programming Interfaces (APIs) allowing consumer electronics manufacturers and 3rd parties to develop applications for the home network. Thus the home network is viewed as a distributed computing platform, and the primary goal of the HAVi Architecture is to assure that products from different vendors can *interoperate* – i.e., can cooperate to perform application tasks.

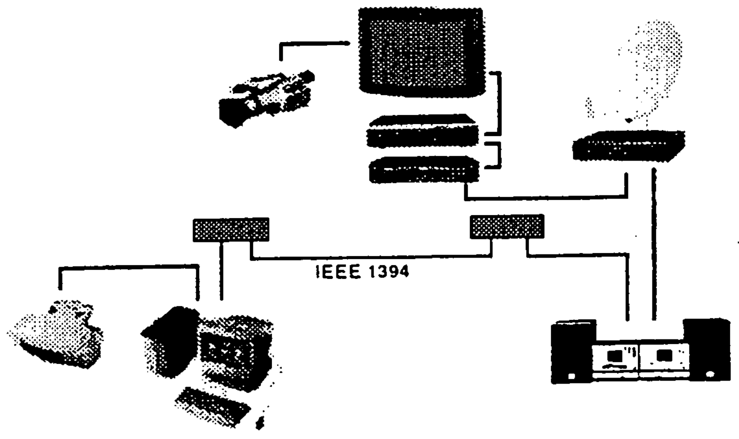
To explain fully the interoperability aspects of the architecture, it is necessary that we begin with an overview of home networking and identify the requirements addressed by the HAVi Architecture.

2.1 The Home Network

Current CE devices, such as DVD players and DV camcorders, are sophisticated digital processing and digital storage systems. By connecting these devices in networks, it is possible to share processing and storage resources – this allows new applications that:

- coordinate the control of several CE devices simultaneously, and
- simplify operation of devices by the user.

For instance one device may initiate recording on a second while accessing EPG information on a third. The home network provides the fabric for connecting CE devices. It allows connected devices to exchange both control information (one device sending a command to another) and AV content (one device sending an audio or video stream to another). To be successful in the consumer electronics domain the home network must meet several requirements. These include: timely transfer of high-data-rate AV streams, self-configuration and self-management, hot plug-and-play, and low-cost cabling and interfaces. The HAVi Architecture is intended for networks based on the IEEE 1394 standard. 1394 is a powerful technology that meets many of the requirements of home networks. An example of a 1394 network is shown below:



The underlying structure for the home network consists of set of interconnected clusters of devices. Typically, there will be several clusters in the home, with one per floor, or per room. Each cluster will work as a set of interconnected devices to provide services to users. Often one device will control other devices. However, the HAVi Architecture is sufficiently flexible to allow home networks with no single master control device.

2.2 Requirements

The HAVi Architecture is an open, light-weight, platform-independent and architecturally neutral specification that allows consumer electronics manufacturers develop interoperable devices and independent application developers to write applications for these devices. It can be implemented on different hardware/software platforms and does not include features that are unique to any one platform.

The interoperability interfaces of the HAVi Architecture are extensible and can be advanced as market requirements and technology change. They provide the infrastructure to control the routing and processing of isochronous and time-sensitive data such as audio and video content.

2.2.1 Legacy Devices Supported

The HAVi Architecture supports legacy devices, i.e. devices that already exist and are available to users. This is important since the transition to networked devices is going to be gradual – with manufacturers not suddenly producing only networked devices and consumers not quickly replacing their existing devices.

Legacy devices can also be characterized by the degree to which they support 1394 and industry standard protocols for 1394 such as IEC 61883. In particular we can divide legacy devices into the following categories:

- non-1394 devices
- 1394 devices not supporting the HAVi Architecture

Most existing CE devices fall into the first category, while existing devices with 1394 interfaces fall into the second category.

HAVi-compliant devices, as opposed to legacy devices, are those that support the HAVi Architecture. The various categories of HAVi-compliant devices are described in section 2.3.3.

2.2.2 Future-Proof Support

The CE industry has great concern that new products work with existing products. While currently this is largely question of media formats and interconnect standards, the HAVi Architecture supports future devices and protocols through several software-based mechanisms. These include:

- persistent device-resident information describing capabilities of devices
- a write-once, run-everywhere language called *HAVi bytecode*
- a device independent representation of user interface elements

Each HAVi-compliant device may contain persistent data concerning its user interface and device control capabilities. This information can include HAVi bytecode that can be uploaded and executed by other devices on the home network. As manufacturers introduce new models with new features they can modify the bytecode shipped with the device. The new functionality added to the bytecode mirrors the new features provided by the device. Similarly new user interface elements can be added to the stored UI representation on the device.

2.2.3 Plug-and-Play Support

Home network consumer devices are easy to install, and provide a significant portion of their value to the consumer without any action on the user's part, beyond physically connecting the cables. This is in distinction to existing devices that require configuration to provide some major portion of their functionality. Home networking technology offers "hot" plug-and-play (not requiring the user to switch off devices), and safe and reliable connections.

In the HAVi Architecture, a device configures itself, and integrates itself into the home network without user intervention. Low-level communication services provide notification when a new device is identified on the network.

While there will often be settings the user may change to suit his or her preferences, the HAVi Architecture does not require the user to perform any additional installation operations (as compared to installation for non-networked or stand-alone usage). Frequently, installing a device on the home network will be simpler than stand-alone installation since new devices can obtain configuration information from those already on the network. Thus the infamous "flashing clock on the VCR" can be solved by having the VCR set its clock to that of another device on the network – for example a DTV receiving time signals via digital broadcast.

2.2.4 Flexible

The HAVi Architecture allows devices to present multiple user interfaces, adapting to both the user's needs and the manufacturer's need for brand differentiation. The architecture includes a flexible device model that scales gracefully from simple CE devices like a CD player or audio amplifier to resource-rich, intelligent devices such as DTV receivers.

2.3 System Model

2.3.1 Control Model

The home network is considered to consist of a set of AV devices. Each device has, as a minimum, enough functionality to allow it to communicate with other devices in the system. The one exception to this are legacy devices (see section 2.3.3).

During the course of interaction, devices may exchange control and data in a peer-to-peer fashion. This ensures that, at the communication level, no one device is required to act as a master or controller for the system. However, it also allows a logical master or controller to impose a control structure on the basic peer-to-peer communication model.

The HAVi control model makes a distinction between *controllers* and *controlled devices*. A controller is a device that acts as a host for a controlled device. A controlled device and its controller may reside on the same physical device or on separate devices.

In terms of the HAVi control model, a controller is said to host a *Device Control Model* (DCM) for the controlled device. The control interface is exposed via the API of this DCM. This API is the only access point for applications to control the device.

For instance, an intelligent television in the family room might be the controller for a number of interconnected devices. Such a controlled devices could contain HAVi bytecode that constructs a user interface for the device and allows external control of the device. When these devices are first connected, the controller obtains the user interface and control code. An icon representing the device may then appear on the television screen, and manipulating the icon may cause elements of the control program to actuate the represented device or devices in prescribed ways.

The home network allows a single device, or a group of devices communicating amongst themselves, to deliver a service to a user or an application. When it is necessary for a device to interact with a user, a GUI for the device may be presented on a device with display capabilities (possibly the device in question or possibly a different device).

2.3.2 Device Model

A distinction is made between *devices* and *functional components*. A good example of this distinction can be found in a normal TV set. Although the TV set is generally one physical box, it contains several distinct entities, e.g. the tuner, audio output etc. The controllable entities within a device are called functional components.

2.3.3 Device Classification

We classify CE devices into four categories: Full AV devices (FAV), Intermediate AV devices (IAV), Base AV devices (BAV) and Legacy AV devices (LAV). HAVi-compliant devices are those in the first three categories, all other CE devices fall into the fourth category.

2.3.3.1 Full AV Devices

A Full AV device contains a complete set of the software elements comprising the HAVi Architecture (see section 2.4.4). This type of device generally has a rich set of resources and is capable of supporting a complex software environment. The primary distinguishing feature of a FAV is the presence of a runtime environment for HAVi bytecode. This allows an FAV to upload bytecode from other devices and so provide enhanced capabilities for their control. Likely candidates for FAV devices would be Set Top Boxes (STB), Digital TV receivers (DTV), general purpose home control devices, and even Home PC's.

2.3.3.2 Intermediate AV Devices

Intermediate AV devices are generally lower in cost than FAV devices and more limited in resources. They do not provide a runtime environment for HAVi bytecode and so can not act as controllers for arbitrary devices within the home network. However an IAV may provide native support for control of particular devices on the home network.

2.3.3.3 Base AV Devices

These are devices that, for business or resource reasons, choose to implement future-proof behavior by providing uploadable HAVi bytecode, but do not host any of the software elements of the HAVi Architecture. These devices can be controlled by an FAV device via the uploadable bytecode or from an IAV device via native code. The protocol between the BAV and its controller may or may not be proprietary. Communication between a Full or Intermediate AV device and a BAV device requires that HAVi commands be translated to and from the command protocol used by the BAV device.

2.3.3.4 Legacy AV Devices

LAV devices are devices that were built before the advent of the HAVi Architecture. These devices use proprietary protocols for their control, and quite frequently have simple control-only protocols. Such devices can work in the home network but require that FAV or IAV devices act as a gateway. Communication between a Full or Intermediate AV device and legacy device requires that HAVi commands be translated to and from the legacy command protocol.

2.4 HAVi Software Architecture

2.4.1 Object-Based

Services in the HAVi Architecture are modeled as objects. Each object is a self-contained entity, called a *software element*, accessible through a well-defined interface and executing within a software execution environment hosted by the device on which the object runs. Note that different devices may host different execution environments. Services are accessed, using the communications infrastructure, via their well-defined interfaces.

Services in the HAVi Architecture can be provided by device manufacturers, or can be added by 3rd party vendors. The software model makes no distinction between "standard" services and vendor services; they are both implemented as objects.

2.4.2 Software Element Identifiers

Each object is uniquely named. No distinction is made between objects used to build system services and those used for application services. All objects make themselves known via a system wide naming service known as the *registry*.

Objects in the system can query the registry to find other objects and can use the result of that query to send messages to those objects.

The identifier assigned to an object is created by the messaging system before an object registers. These identifiers are referred to as SEIDs – *Software Element Identifiers*. SEIDs are guaranteed to be unique, however the SEID assigned to an object may change as a result of reconfiguration of the home network (i.e., device plug-in or plug-out).

2.4.3 Message-Based Communication

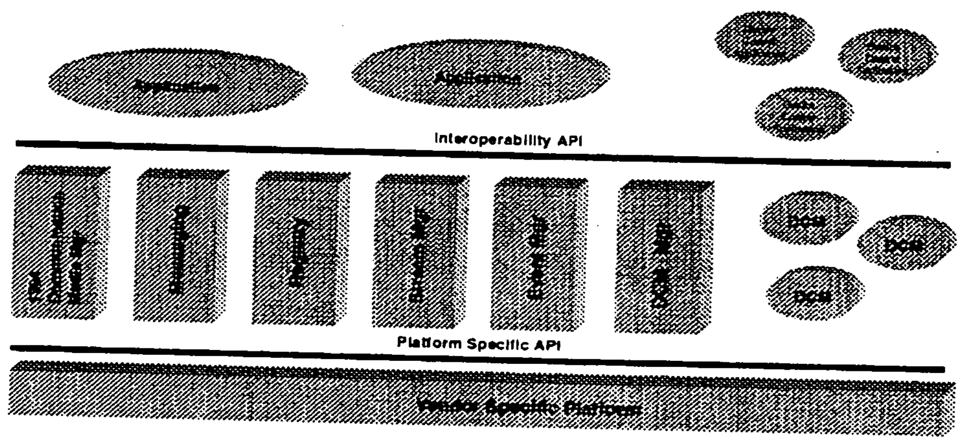
All objects communicate using a message passing model. Any object that wishes to use the service of another object does so by using a general purpose message passing mechanism that delivers the service request to the target object. The target object is specified using the unique SEID discussed above.

This general purpose message passing mechanism abstracts from the details of physical location, i.e. there is no distinction between an object on the same device and one on a remote device. The actual implementation of the message passing infrastructure will differ from device to device and between vendors. However, the format of HAVi messages, and the protocol used for their delivery, must be common so that interoperability is assured.

The general intent of the object model and messaging system is to provide a completely generic software model that is sufficiently flexible to allow multiple implementations with a variety of software systems and languages. Details of the binding between messages and the code that handles them are left to the system implementor.

2.4.4 Software Elements

The software elements of the HAVi Architecture support the basic notions of network management, device abstraction, inter-device communication, and device UI management. Collectively these software elements expose the *Interoperability API*, a set of services for building portable distributed applications on the home network. The software elements themselves reside above a vendor specific platform such as a RTOS. The diagram below depicts the arrangement of software elements on a FAV device. While not intended as an implementation blueprint, the diagram does highlight how the HAVi software elements form a middle layer between platform specific APIs and platform independent applications.



The software elements comprising the HAVi Architecture and defined in this specification are:

- *1394 Communication Media Manager* – allows other elements to perform asynchronous and isochronous communication over 1394.
- *Messaging System* – responsible for passing messages between elements.
- *Event Manager* – serves as an event delivery service. An event is the change in state of an object or of the home network.
- *Stream Manager* – responsible for managing real-time transfer of AV and other media between functional components.

- *Registry* – serves as a directory service, allows any object to locate another object on the home network.

- *Device Control Module* (DCM) – a software element used to control a device. DCMs are obtained from *DCM code units*. Within a DCM code unit are code for the DCM itself plus code for *Functional Component Modules* (FCMs) for each functional component within device. In addition a DCM code unit may include a *device control application* – a software element allowing user control of the device and its functional components.

- *DCM Manager* – responsible for installing and removing DCM code units on FAV and IAV devices.

DCMs are a central concept to the HAVi architecture and the source of flexibility in accomodating new devices and features. DCMs come in two main types:

- *embedded DCM* – a DCM pre-installed on IAV and possibly FAV devices.

- *uploaded DCM* – a DCM implemented in HAVi bytecode. Uploaded DCMs only run on FAV devices.

DCMs can provide APIs for control of both families of devices and specific models. Generally the former APIs will have a wider range of usage, but the latter allow control of vendor-specific features and capabilities.

In addition to the above software elements specified by the HAVi Architecture, devices on the home network may contain the following:

- *HAVi Self Describing Device (SDD) data* – HAVi-compliant devices contain descriptive information about the device and its capabilities. This information, called HAVi SDD data, follows the IEEE 1212 addressing scheme used for Configuration ROM. The HAVi SDD data may include a DCM code unit and data for constructing user interface elements.

- *HAVi Bytecode Runtime* – provides a runtime environment for uploaded DCMs (or applications) implemented using HAVi bytecode.

- *DDI Controller* – a native or bytecode entity involved with user interaction. The DDI (Data Driven Interaction) Controller handles user input and interprets DDI elements.

The following table summarizes which architectural elements are present for the various device categories, which are absent and which are optional. A "check mark" indicates that the element is present on the device itself with the following provisions:

- For both BAV and LAV devices there must be an associated device control module somewhere on the home network. For a BAV device, the DCM is obtained via the device's HAVi SDD data. For LAV devices, the DCM would be pre-installed on the controlling FAV or IAV.

- For IAV and FAV devices it is not necessary that a DCM exists on the home network. (However, if such a DCM does not exist then interoperable applications cannot control the device.)

Device Type / Element	FAV	IAV	BAV	LAV
HAVi Bytecode Runtime	✓			
Stream Manager	✓	[✓]		
DDI Controller	[✓]	[✓]		
DCM Manager	✓	✓		
Registry	✓	✓		
Event Manager	✓	✓		
Messaging System	✓	✓		
1394 Communication Media Manager	✓	✓		
HAVi SDD data	✓	✓	✓	
DCM	[✓]	[✓]	✓	✓

2.5 User Interface Support

The primary goal of the user interface of the home network is to offer a comfortable operating environment to users. The HAVi Architecture allows users to control devices through familiar means, such as via the front panel or via the buttons of a remote controller. In addition the HAVi Architecture allows device manufacturers to specify graphical user interfaces (GUIs) which can be rendered on a range of displays varying from text-only to high-level graphical displays. The GUI need not appear on the device itself, it may be displayed on another device and the display device may potentially be from another manufacturer. To support this powerful feature, the HAVi Architecture introduces a mechanism called *Data Driven Interaction* (DDI). In essence, HAVi SDD data may include *DDI elements* – a platform independent encoding of user interface elements. DDI elements can be loaded and displayed by a *DDI controller*. The DDI controller retrieves the DDI elements via the DCM (rather than directly from the SDD data, so it is possible that the DCM itself is the source of DDI elements). The DDI Controller generates HAVi messages in response to user input, it also responds to events generated by the DCM as a result of changes in device state. This communication is called the *DDI protocol*.

It should be emphasized that the DDI Controller does not understand what happens as a result of issuing a control message. So it is possible to handle new functions which cannot be assumed in advance.

This DDI Controller can not provide guarantees over the graphical rendition of DDI elements, since their representation may be changed due to lack of display screen space or other resource limitations. (Furthermore, application software can create different representations, using the DDI elements as "hints".) However the DDI Controller does try to preserve the appearance of DDI elements subject to its rendering capabilities.

2.5.1 Layout Mechanism

Layout rules of DDI elements are based on geometric coordinates and use x, y values for each DDI element. DDI elements are arranged in a hierarchy and positioned relative to their parents.

The target device *suggests* a preferred layout, which is encoded into the DDI data structure. However, the DDI Controller may tailor the presentation of DDI elements based on its own limitations, such as screen size, ability to display graphics or text only, etc.

2.5.2 Navigation Mechanism

The navigation between DDI elements is handled locally by the DDI Controller. The target device may suggest navigation rules between certain DDI elements. Because such navigation rules are just suggestions, the controller may tailor the navigation of the display based on its adjustment of DDI layout.

2.6 Home Network Configurations

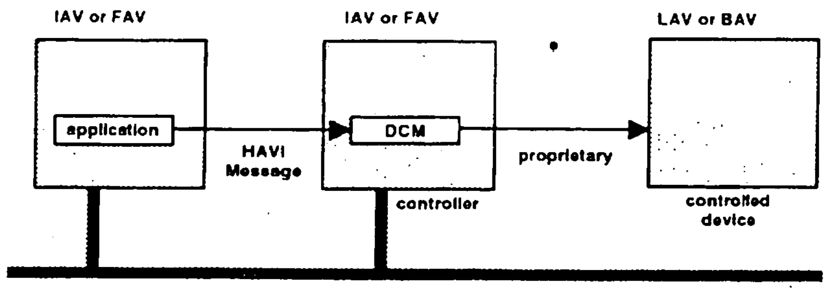
The HAVi Architecture defines the way devices are abstracted within the home network and establishes a framework for device control. It defines the ways that interoperability is assured, and it defines the ways that future devices and services can be integrated into the architecture. The HAVi Architecture makes no restrictions, however, on what types of devices must be present in the home network. As a result several configurations are possible – networks without FAV devices, networks with multiple FAV devices, networks with LAV and BAV devices only, etc. Depending upon the types of devices on the home network, several different operational configurations are possible.

2.6.1 LAV and BAV Only

The HAVi Architecture does not provide any support for networks consisting of only BAV and LAV devices. However with the addition of a HAVi controller (an IAV or FAV) to the network, these devices can be made available to applications.

2.6.2 IAV or FAV as Controller

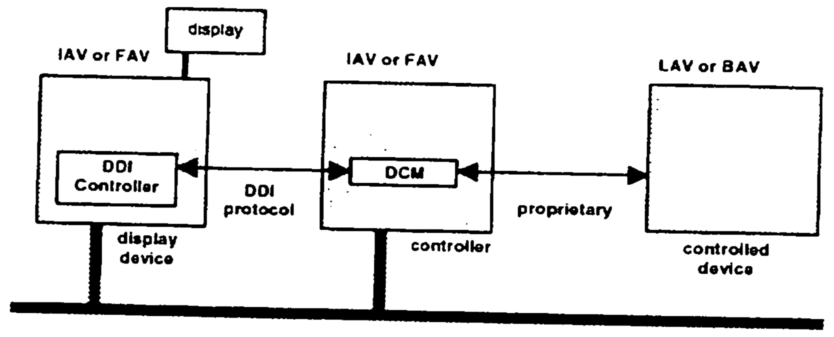
IAV and FAV devices act as controllers for the other device types and provide a platform for the system services comprising the HAVi Architecture. To achieve this FAVs provide a runtime environment for HAVi bytecode DCMs, while IAVs may host embedded DCMs. From an interoperability perspective, the primary role of a controller is to manage DCMs and provide a runtime environment for DCMs. Applications use the APIs provided by the DCM to access the controlled device.



IAV or FAV as controller

2.6.3 IAV or FAV as Display

Generally, IAVs and FAVs will have an associated display device that is used for display of AV content and GUIs. However, the architecture does not mandate this and an IAV or FAV device may be "headless" (without display capability). In this case they will cooperate with other IAV or FAV devices with display capability. Such devices will typically support a DDI Controller. Proprietary low-level graphic manipulation APIs can be used by the DDI Controller to access the display itself, but these interfaces are not exposed as part of the Interoperability APIs.



IAV or FAV as display device

2.6.4 Peer-to-Peer Architecture between FAVs and IAVs

In a home network, there may be more than one FAV or more than one IAV. In this case, each controller (IAV or FAV) cooperates with other controllers to ensure that services are provided to the user. This allows devices to share resources. An example is described in section 2.6.3 where a device without display capabilities uses a remote device to display DCM user interfaces. A more elaborate example could be an FAV device utilizing the services of a data translation module located on a remote device to set up a data stream between two AV devices.

2.6.5 IAV as Controller and Display

A home network may contain no FAV devices, in such cases IAVs are the only entities which may control other devices. Although not equipped with a runtime environment for uploaded DCMs, an IAV may be shipped with a set of *embedded* DCMs. Embedded DCMs can be implemented as native applications on the IAV device and can use native interfaces to access the IAV's display and other resources. Embedded DCMs, like DCMs in general, appear in the registry provided by the HAVi Architecture and can be accessed from other devices on the home network by sending messages over the messaging system. Embedded DCMs, like DCMs in general, may support the DDI protocol and so may participate in providing a user interface for the controlled device.

2.7 Interoperability in the HAVi Architecture

The first and foremost goal of the HAVi Architecture is to support interoperability between AV equipment. This includes existing equipment and future equipment. Because of the need to support existing devices, and because of product cost considerations, the HAVi Architecture supports two levels of interoperability. We refer to these as level 1 and level 2 respectively.

The flexibility of choosing different levels of interoperability is essential in allowing vendors the freedom to design and build devices at all points on the cost/capability spectrum.

2.7.1 Level 1 Interoperability

Level 1 interoperability addresses the general need to allow existing devices to communicate. To achieve this, level 1 interoperability defines and uses:

- a generic set of control messages (commands) that enable one device to talk to another device and
- a set of event messages that it should reasonably expect from the device.

To support this approach a basic set of mechanisms are required.

- *Device discovery*: each device in the home network needs a well-defined method that allows it to advertise its capabilities to others. The approach the HAVi Architecture has adopted is to utilize HAVi SDD data, required on all FAV, IAV and BAV devices. SDD data contains information about the device which can be accessed by other devices. The HAVi SDD data contains, as a minimum, enough information to allow instantiation of an embedded DCM. This results in registration of device capabilities with the HAVi registry, allowing applications to infer the basic set of command messages that can be sent to the device.
- *Communication*: once an application has determined the capabilities of another device, then it needs to be able to access those capabilities. To achieve this requires a general communication facility allowing applications to issue requests to devices. This service is provided the HAVi messaging systems and DCMs. The application sends HAVi messages to DCMs, the DCM then engages in proprietary communication with the device.
- *HAVi message sets*: the last mechanism required to support level 1 interoperability is a well defined set of messages that must be supported by all devices of a particular class. This ensures that a device can work with existing as well as future devices, irrespective of the manufacturer.

These three basic requirements support a minimal level of interoperability. Since any device can query the capabilities of another via the registry, any device can determine the message set supported by another device. Since applications have access to the messaging system, then any device can interact with any other device.

2.7.2 Level 2 Interoperability

Level 1 interoperability ensures that devices can interoperate at a basic level of functionality. However, a more extensible mechanism is also needed to allow a device to communicate to other devices any additional functionality not present in embedded DCMs. For example, embedded DCMs may not support all features of existing products and are unlikely to support future product categories. Level 2 interoperability provides this mechanism.

To achieve this, the HAVi Architecture allows uploaded DCMs as an alternative to embedded DCMs. The uploaded DCM may replace an existing DCM on FAV devices. The HAVi Architecture makes no statement about the source of the uploaded DCM, but a likely technique is to place the uploaded DCM in the HAVi SDD data of the BAV device, and upload from the BAV to the FAV device when the BAV is attached to the home network. Because the HAVi Architecture is vendor neutral, it is necessary that the uploaded DCM will work on a variety of FAV devices all with potentially different hardware architectures. To achieve this, uploaded DCMs are implemented in HAVi bytecode. The HAVi bytecode runtime environment on FAV devices supports the instantiation and execution of uploaded DCMs.

Once created and running within a FAV device, the DCM communicates with the LAV and BAV devices in the same manner as described above in section 2.7.1.

The efficiency of level 2 interoperability appears when one considers resources needed to access device functionality. Level 2 allows a device to be controlled via an uploaded DCM that presents all the capabilities offered by the device. Whereas to achieve similar functionality in level 1, this DCM would have to be embedded somewhere in the network. For example when a new device is added to a network, level 1 requires that at least one other device contains a DCM suitable for the new device. In comparison, level 2 only requires that one device provide a runtime environment for the uploaded DCM obtained from the new device.

The concept of uploading and executing bytecode also provides the possibility for device specific applications called *Device Control Applications*. By these applications a device manufacturer can provide the user a way to control special features of a device without the need for standardizing all the features in HAVi. The application is provided by a DCM in HAVi bytecode and can be uploaded an installed by each FAV device on the network.

2.8 Versioning

Each HAVi component must support the HAVi version control API. HAVi version control is intended to maintain interoperability of HAVi components as the specification evolves. Version control for individual manufacturer's products is outside of the scope of this API.

Versions are represented by major and minor numbers in the form `major.minor`. For a given release of the specification, all components will be of the same version as that of the HAVi specification - regardless of whether that component's API was modified in the latest release. This simplifies the situation in which a given component's APIs are built up from different groups of APIs. The minor version number is intended indicate small refinements of the HAVi specification. The intent of the major version number is to reflect important functional improvements in the overall HAVi architecture.

Given that this draft specification is Version 0.8, all HAVi components are currently defined to be at Version 0.8. It is intended that no APIs will be changed or removed as the draft specification evolves, though this is not absolutely guaranteed. Updates to existing APIs will be generally be achieved by adding new APIs. (e.g. if `int getFoo()` changes during the evolution of the draft specification, `int getFooA()` may be added to the API, and the original `int getFoo()` will not be removed.)

At the first official release of the specification, all components will be of Version 1.0. At this time all deprecated APIs will be removed from the specification, and the updated APIs will be renamed. (`int getFooA()` would be renamed `int getFoo()` and the older `int getFoo()` API would be removed).

As the official release of the specification evolves, it is intended to no APIs will ever be updated or removed. All changes will be achieved by adding new APIs. This ensures that older components can always request services from newer components successfully.

Before a component requests a service, it must first check the version of the component which is to provide that service. If the version of the component providing the service is greater than the version of the component which is requesting the service, the requesting component must communicate with the service provider using the requestor's most current APIs.

If, on the other hand, the requestor finds that the service provider is of an older version, the requestor must restrict itself to APIs that are known to be supported by the older component. A minimum requirement is that the requestor must be able to interact with older providers using protocols from all major releases of the specification older than the requesting component. As an example, if Component A33 is of Version 3.3, Component B24 is of Version 2.4 and Component C12 is at Version 1.2, A33 is required to communicate with B24 using APIs of Version 2.0 at a minimum. When A33 or B24 communicate with component C12, the minimum API version is 1.0. The manufacturer may also choose to differentiate between Versions 2.1, 2.2, 2.3 and 2.4(in the case of A33 requesting a service from B24), but this is optional.

The rules for version control of message passing protocols is somewhat different. Please refer to the Message Passing section for details.

Applications are encouraged to support older component versions, though this is not explicitly required. This is intended to ensure that interoperability will be achieved for the life of the HAVi specification.